PrAMC-7311 BBS based on Ubuntu 16.04.01 LTS

Programmer's Reference

P/N: 6806800T73B

November 2019

SMART™ Embedded Computing

Trademarks

The stylized "S" and "SMART" is a registered trademark of SMART Modular Technologies, Inc. and “SMART Embedded Computing” and the SMART Embedded Computing logo are trademarks of SMART Modular Technologies, Inc. All other names and logos referred to are trade names, trademarks, or registered trademarks of their respective owners. These materials are provided by SMART Embedded Computing as a service to its customers and may be used for informational purposes only.

Disclaimer*

SMART Embedded Computing (SMART EC) assumes no responsibility for errors or omissions in these materials. **These materials are provided "AS IS" without warranty of any kind, either expressed or implied, including but not limited to, the implied warranties of merchantability, fitness for a particular purpose, or non-infringement.** SMART EC further does not warrant the accuracy or completeness of the information, text, graphics, links or other items contained within these materials. SMART EC shall not be liable for any special, indirect, incidental, or consequential damages, including without limitation, lost revenues or lost profits, which may result from the use of these materials. SMART EC may make changes to these materials, or to the products described therein, at any time without notice. SMART EC makes no commitment to update the information contained within these materials.

Electronic versions of this material may be read online, downloaded for personal use, or referenced in another document as a URL to a SMART EC website. The text itself may not be published commercially in print or electronic form, edited, translated, or otherwise altered without the permission of SMART EC.

It is possible that this publication may contain reference to or information about SMART EC products, programming, or services that are not available in your country. Such references or information must not be construed to mean that SMART EC intends to announce such SMART EC products, programming, or services in your country.

Limited and Restricted Rights Legend

If the documentation contained herein is supplied, directly or indirectly, to the U.S. Government, the following notice shall apply unless otherwise agreed to in writing by SMART Embedded Computing.

Use, duplication, or disclosure by the Government is subject to restrictions as set forth in subparagraph (b)(3) of the Rights in Technical Data clause at DFARS 252.227-7013 (Nov. 1995) and of the Rights in Noncommercial Computer Software and Documentation clause at DFARS 252.227-7014 (Jun. 1995).

SMART Embedded Computing, Inc.

2900 S. Diablo Way, Suite 190

Tempe, Arizona 85282

USA

*For full legal terms and conditions, visit *www.smartembedded.com/ec/legal*

Table of Contents

Table of Contents

List of Figures

List of Figures

List of Tables

List of Tables

About this Manual

Overview of Contents

This manual is divided into the following chapters and appendices.

Abbreviations

This document uses the following abbreviations:

Abbreviation	Definition
AMC	Advanced Mezzanine Card
ATCA	Advanced Telecommunications Computing Architecture
BBS	Basic Blade Services
BIOS	Basic Input Output System
DHCP	Dynamic Host Configuration Protocol
FCU	FUF Command Line Utility
FRI	Firmware Recovery Image
FRU	Field Replaceable Unit
FUF	Firmware Upgrade Facility
GPIO	General Purpose Input/Output
HPI	Hardware Platform Interface
HPM	Hardware Platform Management
IPMB	Intelligent Platform Management Bus
IPMC	Intelligent Platform Management Controller

Abbreviation	Definition
IPMI	Intelligent Platform Management Interface
LUN	Logic Unit Number
MAC	Media Access Control
MMC	Module Management Controller
PCI	Peripheral Component Interconnect
PCIEX	PCI Express
PICMG	PCI Industrial Computers Manufacturers Group
PXE	Preboot Execution Environment
RMCP	Remote Monitoring and Control Protocol
RTM	Rear Transition Module
SAF	Service Availability Forum
SATA	Serial ATA
SDR	Sensor Data Record
SMI	Serial Management Interface
SNMP	Simple Network Management Protocol
SOL	Serial Over Lan
SSH	Secure Shell
TAR	Tape Archiver
TFTP	Trivial File Transfer Protocol
UDP	User Datagram Protocol

Conventions

The following table describes the conventions used throughout this manual.

Notation	Description
`0x00000000`	Typical notation for hexadecimal numbers (digits are 0 through F), for example used for addresses and offsets
`0b0000`	Same for binary numbers (digits are 0 and 1)
bold	Used to emphasize a word
`Screen`	Used for on-screen output and code related elements or commands. `Sample of Programming used in a table (9pt)`
`Courier + Bold`	Used to characterize user input and to separate it from system output
Reference	Used for references and for table and figure descriptions
File > Exit	Notation for selecting a submenu
<text>	Notation for variables and keys
[text]	Notation for software buttons to click on the screen and parameter description
...	Repeated item for example node 1, node 2, ..., node 12
. . .	Omission of information from example/command that is not necessary at the time
..	Ranges, for example: 0..4 means one of the integers 0,1,2,3, and 4 (used in registers)
|	Logical OR
	Indicates a hazardous situation which, if not avoided, could result in death or serious injury
	Indicates a hazardous situation which, if not avoided, may result in minor or moderate injury

Notation	Description
	Indicates a property damage message
	Indicates a hot surface that could result in moderate or serious injury
	Indicates an electrical situation that could result in moderate injury or death
	Indicates that when working in an ESD environment care should be taken to use proper ESD practices
	No danger encountered, pay attention to important information

Summary of Changes

Part Number	Date	Description
6806800T73B	November 2019	Rebrand to SMART Embedded Computing template. Updated Ubuntu version to 16.04.01; Trusty Tahr updated to Xenial Xerus; ubuntu-trusty updated to ubuntu-xenial; HPM and FUF Installation folder name/directory updated; kernel version updated; Table 2-1 documentation file names updated.
6806800T73A	October 2015	Initial version.

Introduction

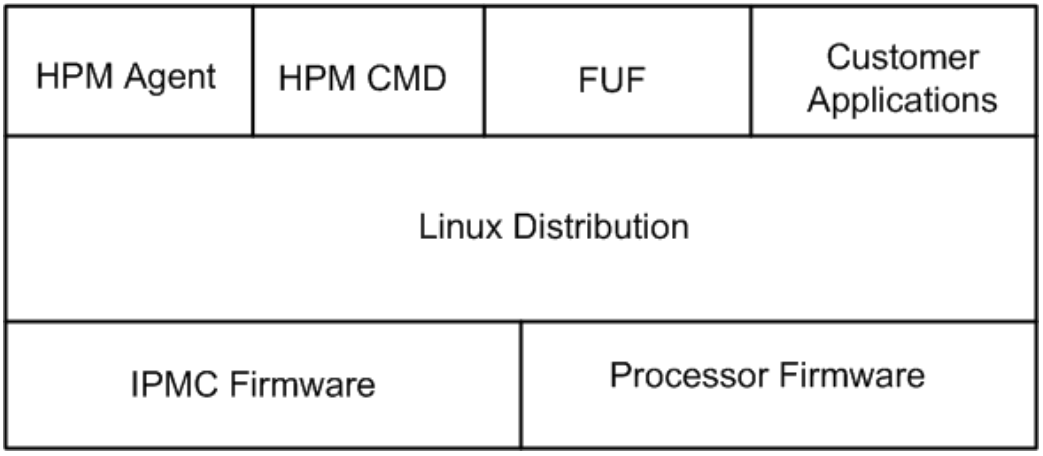
The Basic Blades Services (BBS) software provides a set of services that support the module on which the software is installed.

1.1 Software Building Blocks

BBS services include a common set of functionality which is available for all AdvancedTCA blades and Advanced Mezzanine Card (AMC) modules, and a unique set of functionality which is tailored to a particular blade or module.

The figure below depicts the architecture of the BBS software.

Figure 1-1 BBS Architecture



BBS consists of the following software and services:

- Firmware Upgrade Facility (FUF)
  The Firmware Upgrade Facility provides a uniform way to upgrade firmware on SMART Embedded Computing blades and AMC modules, regardless on which flash locations the firmware is stored. FUF upgrades the Basic Input Output System (BIOS) firmware as well as the IPMC firmware. The FUF currently consists of a Firmware Upgrade Command Line Utility (FCU), flash device drivers, and supports specially prepared Firmware Recovery Image (FRI) files as well as HPM.1 compatible firmware images. The FUF can be used on switch and node blades and on AMC modules.

- Hardware Platform Management (HPM)
  HPM in AdvancedTCA systems is based on Intelligent Platform Management Interface (IPMI) specification. IPMI commands can be complex and cumbersome. Using a certain set of commands, HPM facilitates the blade or module-level hardware management.
- HPM Agent (HPMA)
  The HPM agent daemon handles local communication to the Intelligent Platform Management Controller (IPMC) on a module using the Serial Management Interface (SMI). It invokes shutdown or reboot scripts based on received IPMI requests.

Installing the Basic Blade Services

Software

SMART Embedded Computing provides software images, including software updates, to its licensed customers. In order to obtain the latest BBS software, please contact your local sales representative.

2.1 Package Information

BBS software is packaged as a `.tar.gz` image. In general, you need to uncompress the image to install the packages. The following table describes about each directory in the image.

Table 2-1 BBS Package Information

Directory	Files	Distribution
BBS-PKGS	SUBDIR: `pramc7311_ga/bbs-pkgs/debs`	
	`bbs-fcu_<version>_amd64.deb`	FUF package is to upgrade the blade's BIOS, FPGA and the IPMC firmware on the front blade and the RTM.
	`bbs-hpmcmd_<version>_amd64.deb`	Package provides a set of commands that are based on IPMI commands but which are easier to use.
	`bbs-hpmagent_<version>_amd64.deb`	Package handles the events from IPMC.

Table 2-1 BBS Package Information (continued)

Directory	Files	Distribution
HPI-B	SUBDIR: HPI-B_Pkgs	
	`bbs-hpib_<version>_amd64.deb`	Package includes Shared libraries to be used by HPI-B clients and some example HPI-B client applications.
	`bbs-hpib-devel_<version>_amd64.deb`	Package includes HPI-B header files and static libraries used for the development of HPI-B clients. This is part of host development toolkit.
	`bbs-hpib-clientsrc_<version>_amd64.deb`	Package includes Client sources as part of devkit for the application. This is part of host development toolkit.
	SUBDIR: HPI-B_Multishelf_Sources	
	`bbs-hpib-<version>.tar.gz`	This zip file contains the HPI-B client library sources.
LSP	Kernel Modules:	
	`lib/modules/<kernel version>/kernel/drivers/char/ipmi/ipmi_poweroff.ko`	`ipmi_poweroff` module with modification to handle AMC shutdown from shelf manager.
	`lib/modules/<kernel version>/kernel/drivers/net/ethernet/intel/e1000e/e1000e.ko`	Intel 82571 driver with modification for handling the slow notification of the interface status to a bonding interface. The time for interface to come up is reduced to less than 1 second.
	SUBDIR: patches	
	`ipmi_poweroff.patch`	Patch file for generating `ipmi_poweroff` changes generated on Ubuntu linux source
	`e1000e-3.2.4.2.patch`	Patch file for generating modified Intel 82571 driver generated on e1000e-3.2.4.2 source code base.

Table 2-1 BBS Package Information (continued)

Directory	Files	Distribution
FIRMWARE	`pramc-7311-cpu_<version>.fri`	BIOS Firmware image to upgrade the BIOS module.
	`pramc-7311-ipmc-<version>.hpm`	IPMC Firmware image to upgrade the IPMC module.
	`pramc-7311-ipmc-boot-<version>.hpm`	IPMC Boot loader Firmware image to upgrade the IPMC Boot Loader module.
DOCUMENTATION	`PrAMC_7311_BBS_RN_UBUNTU_16.04.01_<version>.pdf`	Release notes.
	`PrAMC_7311_BBS_PR_UBUNTU_16.04.01_<version>.pdf`	BBS programmers guide.

2.2 Accessing the Serial Console

To access the PrAMC-7311 via serial console, the default serial port or terminal emulator settings are as follows:

- 115200 baud
- No parity
- Eight data bits
- One stop bit

2.3 Installation

For installing/booting the Ubuntu BBS software on the PrAMC-7311, we need to install the Ubuntu 16.04.01 LTS distribution first using the diskless client boot via network method, followed by the installation of required BBS package using apt/dpkg tools.

For installing the Ubuntu 16.04.01 LTS using diskless client method, you need to set up an external TFTP server to retrieve the required kernel image and rootfs image for installation. Furthermore you need to do some initial configurations. The following table provides main setup and configuration steps of these installation/boot options. The detailed procedures can be found in the following sections.

Table 2-2 Ubuntu Installation/Options - Main Set-Up and Configuration Steps

Installation/Boot Option	Main Set-Up and Configuration Steps
Diskless client boot	1. Set up and configure external TFTP boot server 2. Configure DHCP server 3. Configure PXE boot options 4. Configure PrAMC-7311 BIOS to boot from network

2.3.1 Prerequisites

The following files are required for the installation of Ubuntu on PrAMC-7311.

Table 2-3 Ubuntu Ramdisk and Kernel Image Files

File	Description
`linux`	Ubuntu 16.04.01 kernel image
`initrd.gz`	initramdisk image containing the Ubuntu root file system
`pxelinux.0`	Pxelinux pre boot loader image

These files can be obtained from the official Ubuntu 16.04.01 LTS Xenial Xerus distribution ISO file. Mount the ISO file using the following command:

```
mount –t iso9660 –o loop ubuntu-16.04.01-server-amd64.iso ubuntu_mount
```

You can find the required files mentioned in the above table for network installation in the `ubuntu_mount/install/netboot/ubuntu-installer/amd64/` folder. These files are needed to be copied/mounted in the tftpboot directory for diskless booting.

Another prerequisite for the installation of Ubuntu OS is the availability of the non volatile storage space. If you are using the SATA HDD mounted on ATCA-F140, make sure that the HDD is properly routed to PrAMC-7311 by running `hpm muxctrl` commands in ATCA-F140.

Example:

```
hpmcmd -c muxctrl mux15 B2
```

2.4 Configuring TFTP, DHCP and PXE

The following procedures assume that the required files are available on a TFTP server for the Ubuntu installation. For all installation and boot options, you need to set up and configure a TFTP server. Furthermore, for the diskless client and hard disk installation/boot option, you need to configure the system's DHCP server and configure PXE boot options. All related steps are described in the following section.

2.4.1 Setting Up and Configuring the TFTP Server

It is customary to place TFTP files in a `/tftpboot` directory. Regardless of the file system node you specify as the root for your TFTP service, the installation scripts expect a certain directory structure when retrieving files.

Creating the /tftpboot Directory and Copying the Target Files

To create the expected directory structure and copy the needed files, perform the following:

1. On the host, create a `/tftpboot` directory if it does not already exist.
   `mkdir /tftpboot`
2. Copy the required files for your installation to the `/tftpboot` directory. Depending on the installation/boot option, the required files are the following.

Installation/Boot	File	Description
Diskless client	`linux`	Linux kernel to start Ubuntu installation
	`initrd.gz`	Initramdisk image for Ubuntu installation
	`pxelinux.0`	prelinux preboot image

Now, configure the TFTP server. The procedure assumes that the TFTP server is under the control of xinetd. For any other servers, this procedure will be different. It is essential that your server provides TFTP support and that the files are copied to the directories as specified in *Setting Up and Configuring the TFTP Server on page 21*.

Configuring a TFTP Server

To configure TFTP as root on the host, perform the following steps:

1. Create (or edit) the file `/etc/xinetd.d/tftp`. This file might be located somewhere else in your system, therefore, the first line might be different. Use the following format:

```
#/etc/xinetd.d/tftp
service tftp
{
   socket_type = dgram
   wait = yes
   user = root
   log_on_success += USERID
   log_on_failure += USERID
   server = /bin/in.tftpd
   server_args = -r blksize /tftpboot
   disable = no
   protocol = udp
}
```

2. Create the directory `/tftpboot` and add needed files as described in *Setting Up and Configuring the TFTP Server on page 21*.
3. If there are any TFTP daemons that have not timed out, you need to stop them. Enter the following command to do so:
   `killall in.tftpd`
4. Enter the following command to have **`xinetd`** re-read its configuration file:
   `/etc/rc.d/init.d/xinetd restart`

Your TFTP server is now configured.

2.4.2 Configuring DHCP

In order to configure DHCP, you need to edit the configuration file of the DHCP server. The DHCP configuration file on an TFTP server resides in `/etc/dhcpd.conf.` file, you need to provide the Media Access Control (MAC) and IP address of the PrAMC-7311.

One way to determine the MAC address of the interface on your PrAMC-7311 is by letting BIOS attempt to PXE boot. The screen will show, for example:

```
Intel (R) Boot Agent GE v1.2.40
Copyright (C) 1997-2006, Intel Corporation

CLIENT MAC ADDR: 00 01 AF 18 F2 68
```

In this example the MAC address would be 00:01:AF:18:F2:68.

The configuration file should look similar to the following extract.

```
#
# DHCP server sample configuration file
#
```

```
allow booting;
allow bootp;

#
#   A declaration for each subnet in your network that requires DHCP
service
#   Note that your DHCP server needs an interface addressed to each of
these #   subnets
#   These declarations can be used to define options common to all blades
in
#   that subnet
#

subnet 192.168.1.0 netmask 255.255.255.0
{
    option broadcast-address 192.168.1.255;
    option routers 192.168.1.253;
}

<Additional subnet declarations as required>

#
#  Blade declarations. One for each blade requiring DHCP service
#  Blades are identified by their ethernet address
#  7201 (and other boards using BIOS) should use filename "pxelinux.0"
#       and will have further configuration information contained in
#  /tftpboot/pxelinux.cfg
#  6201 (and other boards using MotLoad) do not need a filename
#        as it can be specified in the individual board's firmware boot
command
#

host 7311_blade1
{
    hardware ethernet 00:01:AF:18:F2:68;
    fixed-address 192.168.1.1;
    filename "pxelinux.0";
}


<Additional blade declarations as required>
```

On Linux servers, the DHCP service is usually started with:
#/etc/init.d/dhcpd start

Make sure the DHCP daemon is running:
 # ps -aef|grep dhcpd

The following is displayed if the daemon is running:
.../usr/bin/dhcpd...

Restart the daemon to apply the changes made to the DHCP configuration file:
 # /etc/init.d/dhcpd restart

2.4.3 Configuring PXE

Once BIOS obtains an IP address using DHCP, it will download and execute the file `pxelinux.0` configured in the DHCP entry. This utility will return to the server to get further information required to boot the Linux kernel. It looks for the below information in the `/tftpboot/pxelinux.cfg` directory.

First, it will check for a file name using the Ethernet hardware address, all in lower case hexadecimal with dash separators; the file name has the `01-<MAC address>` format. For example, if the Ethernet address is 00:01:AF:17:5E:90 it would search for the file name 01-00-01-af-17-5e-90.

Next, it looks for a file name based on its assigned IP address in upper case hexadecimal. For example, if the assigned IP address is 192.168.1.1, it will check for a file named C0A80101. If that file is not found, it will remove one trailing hexadecimal digit of the hexadecimal IP address file name and iteratively try down to the first digit (C0A8010, then C0A801, then C0A80, ... C).

If none of these module-specific files exist, it will look for a file named **default**. Module-specific files allow each module (or a group of modules) to boot using different kernel and file system images. The contents of any of these files follows a similar format:

```
DEFAULT <kernel file name> initrd=<file system image name>
console=ttyS0,<serial configuration> root=/dev/ram0 <other options>
```

where:

`<kernel file name>` includes the path along with the kernel file name in the `/tftpboot` directory. For example, in the `/tftpboot` location, if there is a PrAMC-7311 directory that contains the kernel image *linux*, the `<kernel file name>` is `/PrAMC-7311/linux.`

`<file system image name>` includes the path along with the system image file in the `/tftpboot` directory. For example, in the `/tftpboot` location, if there is a PrAMC-7311 directory that contains Linux root file system image *initrd.gz*, then the `<file system image name>` is `/PrAMC-7311/initrd.gz`.

<other options> are any additional desired Linux command line options.

For a diskless PrAMC-7311installation of Ubuntu, the file would normally contain:

```
DEFAULT /PrAMC-7311/linux initrd=/PrAMC-7311/initrd.gz
console=ttyS0,115200n8 root=/dev/ram0
```

2.5 Installation Procedures

The below installation procedures are described in the following subsections.

- Diskless Client Network Boot Installation
- BBS Software Installation

2.5.1 Diskless Client Network Boot Installation

The PrAMC-7311 can be booted as diskless client from a boot server using `pxeboot`. In order to configure the PrAMC-7311 accordingly, the following preparative steps are required.

Configuring BIOS and Reboot

To configure BIOS, proceed as follows.

1. Connect to the PrAMC-7311 via the serial console interface (see *Accessing the Serial Console on page 19*).
2. Power up or reboot the PrAMC-7311 and hold down the <F2> key on your keyboard until the BIOS menu appears.
3. Select `Boot` on the top menu.
4. Select the network boot device. Depending on your system configuration, this device is named similar to the following: `PCI LAN: IBA GE Slot 010x v1381`
   Make sure that this device is enabled, that is, it is not part of the "Excluded from boot order" list.
5. Move the network boot device with the <+> key so it is the first enabled device.
6. Save and exit.
   It takes about 30 seconds to reboot.
7. While the module is booting, observe that it is getting a DHCP address and then loading the kernel and initrd image:
   ```
   Trying to load: pxelinux.cfg/00000000-0000-0000-0000-000000000000
   Trying to load: pxelinux.cfg/xx-xx-xx-xx-xx-xx
   ```

```
Trying to load: pxelinux.cfg/0AF741BB
boot:
Loading PrAMC-7311/linux.........................
Loading PrAMC-7311/initrd.gz...................
```

8. Once the kernel boots up, it appears on the installation menu with keyboard selectable options from where you can continue the Ubuntu installation as usual. Ensure that the front panel ethernet interface is connected to internet as it needs to download the required packages during the course of installation. Need to configure the correct Ubuntu repositories during installation to get the packages installed.
9. Once the installation is done and rebooted to the OS, you can add more packages using the apt-get tools.

2.5.2 LSP Installation

To install the provided e1000e and ipmi_poweroff kernel modules, refer *Update initramfs image on page 29*.

2.5.3 BBS Software Installation

2.5.3.1 HPM and FUF Installation

The /pramc7311_ga/bbs_pkgs/ folder contains software for HPM, HPMAgent and FUF modules. Copy this directory to the target PrAMC-7311 to any suitable location for installation.

Before installing these modules, install openipmi package using the following apt-get command:

```
# apt-get install openipmi
```

To install HPM and FUF, execute the following steps:

1. Change to /pramc7311_ga/bbs_pkgs/ directory.
   For example:
   ```
   # cd pramc7311_ga/bbs_pkgs
   ```
2. Run the shell script install.sh to install the BBS utility packages:
   ```
   # ./install.sh
   ```
3. After completing the installation, add /opt/bladeservices/bin to PATH environment variable or logout and login again:
   ```
   # export PATH=$PATH:/opt/bladeservices/bin
   ```

To uninstall HPM and FUF, execute the following shell command:

```
# uninstall in /opt/bladeservices/bin
```

Linux Distribution Description

3.1 Distribution Description

The BBS software for the PrAMC-7311 is based on Ubuntu 16.04.01 and kernel version 4.8.0-26-generic.

3.2 Login

A Linux shell can be accessed via the face plate serial port.

If you use a serial console or terminal emulator, the serial port settings are as follows:

- 115200 baud
- No parity
- Eight data bits
- One stop bit

If you use Secure Shell (SSH), see *Network Services Configuration on page 27* for default IP address assignments.

Login to the terminal using the default username/password created during the installation time. If the root password is not set, you can set it using the `passwd` command once you are logged in as default user `sudo passwd`.

3.3 Network Services Configuration

The following Ethernet interfaces are configured at startup:

Interface	IP Address
p1p1	Front Panel
p1p2	Base1
eth1	Base2
p1p3	Fabric1
eth2	Fabric2

3.4 Installing Linux Kernel Patches

The directory LSP of the release folder contains patch sources for e1000e and ipmi_poweroff modules and also their corresponding module binaries maintained in the directory structure identical to the ubuntu root file system.

Example:

The modified kernel module for ipmi_poweroff, will be made available in the `lsp/lib/modules/4.8.0-26-generic/kernel/drivers/char/ipmi/`

3.4.1 Compiling the modules by applying their respective patches

3.4.1.1 e1000e

1. Download the source code for `e1000e-3.2.4.2.tar.gz`. It can be downloaded from the location *http://sourceforge.net/projects/e1000/files/e1000e%20stable/3.2.4.2/*.
2. Untar the content using
   `$> tar -xzf e1000e-3.2.4.2.tar.gz`
3. Run patch command
   `$> patch -p0 < e1000e-3.2.4.2.patch`
4. Change to `e1000e-3.2.4.2/src directory`.
   `$> cd e1000e-3.2.4.2/src`
5. Build the driver. The module named `e1000e.ko` will be available in the `src` folder.
   `$> make -C /lib/modules/`uname -r`/build M=`pwd` modules`
6. Use **`rmmod e1000e; insmod e1000e.ko`** commands for removing the existing and inserting the new module.

3.4.1.2 ipmi_poweroff

1. Run **`git clone`** *http://kernel.ubuntu.com/git-repos/ubuntu/ubuntu-xenial.git* in some directory where you want to download the source code for the running kernel. This will create **`ubuntu-xenial`** folder.
2. Change into **`ubuntu-xenial`** folder and run
   git checkout tags/Ubuntu-lts-4.8.0-26_16.04.1
3. Copy the `'ipmi_poweroff.patch'` to **`ubuntu-xenial`** folder.
4. Change to **`ubuntu-xenial`** directory and run
   `patch -p0 < ipmi_poweroff.patch.`
5. **`cd drivers/char/ipmi folder`**

6. **`make -C /lib/modules/`uname -r`/build M=`pwd``** modules for building the driver.
7. **`rmmod ipmi_poweroff, insmod ipmi_poweroff.ko`** commands for removing the existing and inserting the new module.

NOTE: The `ipmi_poweroff.ko` module is not loaded by default in Ubuntu across the reboots. For the deactivation/shutdown of the PrAMC-7311 using hpmagentd to work properly, we need the `ipmi_poweroff` module loaded as well. So, ensure that `ipmi_poweroff` module is inserted before performing deactivation. Ensure to load the modules across the reboot by modifying the Ubuntu configuration file `/etc/default/openipmi` with `IPMI_POWEROFF=yes`.

3.4.2 Update initramfs image

To make the generated modules part of initramfs, copy the module binaries to their corresponding directory in `/lib/modules/<kernel version>` folder.

Example:

copy **`e1000e.ko to /lib/modules/4.8.0-26-generic/kernel/drivers/net/ethernet/intel/e1000e`** and run **`update-initramfs -u`**

Firmware Upgrade Facility

The Firmware Upgrade Facility (FUF) provides a uniform way to upgrade firmware on SMART EC hub blades, node blades, and AMC modules. It consists of a Firmware Upgrade Command-line Utility (FCU), flash device drivers, and specially prepared firmware recovery image files.

4.1 Firmware Recovery Image Files

The PrAMC-7311 supports the HPM.1 IPMI standard. FCU works with specially prepared firmware recovery image (FRI) files or with HPM.1 compatible files. The following firmware packages are currently available:

Filename	Description
`pramc-7311-cpu_<version>.fri`	BIOS firmware image for PrAMC-7311
`pramc-7311-ipmc-boot-<version>.hpm`	IPMC bootloader
`pramc-7311-ipmc-<version>.hpm`	IPMC firmware

4.2 Backup Concept

The BIOS firmware for the PrAMC-7311 is stored in redundant, persistent memory devices. This allows the firmware image in one bank to serve as a backup for the other bank. This is particularly useful for firmware upgrades.

During normal operation, the CPU determines which bank to boot from based on a chip select signal controlled by the IPMC. This bank is considered the active boot device. FCU will only allow you to upgrade an inactive device. It determines whether a device is active or inactive by querying the IPMC to learn which device is marked to be used at boot. Because you can change the “active” device with the FCU mark operation, active status does not necessarily indicate which device was used on the last boot. It simply represents which device is set for use on the next boot.

The IPMC firmware consists of a boot loader as well as an active and a stand-by IPMI firmware. The boot loader maintains both the active and stand-by firmware in the flash memory of the PrAMC-7311. Both the boot loader as well as the IPMI firmware images can be upgraded.

Each time the IPMC firmware is upgraded, the most recent firmware version is kept in flash memory and the older firmware version is overwritten by the new one. Once the new IPMI firmware is programmed, the IPMC resets itself to boot from the new image. The boot loader validates the new IPMC firmware. Provided the IPMC can power up successfully the current image is made active and the previously active image is made backup. In case of power-up failures, the boot loader automatically recovers from crisis and boots from the previous image.

4.3 fcu – Firmware Upgrade Command-Line Utility

The Firmware Upgrade Command-line Utility (FCU) allows you to:

- Query the current versions of firmware installed on a module and determine which firmware devices are active
- Verify that a specified upgrade image is sound and compatible with the current hardware
- Upgrade a firmware image
- Mark a device to be used as the boot source on the next reset
- Display detailed information about a firmware upgrade image file

By default, the FCU binary executable is installed in `/opt/bladeservices/bin`. This directory has been added to the PATH environment variable.

FCU works in conjunction with device drivers created specifically for the flash devices on SMART EC modules.

The FCU verify and upgrade operations require specially prepared FRI files or HPM.1 compatible image files. For more information, see *Firmware Recovery Image Files on page 31*.

Synopsis

```
fcu --help [-t<slave address>]
fcu --version
fcu -q [-d <device-id>]
fcu -v -f <filename>
fcu -u -f <filename>
fcu -a -f <filename>
fcu -s <filename>
fcu -m -b <bank-letter> -d <device-id>
```

Parameters

`-b <bank-letter>`
`--bank=<bank-letter>`

Specifies the flash bank (For example: A/B or 0/1), to mark for next boot, where <bank-letter> is the letter designating a specific bank. This option is used with the mark operation. Use the query option -q to list available banks.

`-d <device-id>`
`--device=<device-id>`

Specifies a target firmware device, where <device-id> is the name of the device. This option is used with the mark or query operations. Device ID values vary by hardware. You can display supported devices on a given module by using **`fcu --help`**. Currently supported values are listed in the following table.

Device ID	Description
pramc-7311-cpu	CPU firmware device for PrAMC-7311
H8S-AMCm F/W	IPMC firmware device for PrAMC-7311
H8S-AMCm B/L	IPMC boot loader device for PrAMC-7311

`-f <filename>`
`--file=<filename>`

Specifies the FRI or HPM.1 image file, where <filename> is the complete path and filename of the file. This option is used with the verify and upgrade operations.

`--help`

Displays a brief message describing command usage. It also displays a list of the devices supported on the module.This option is exclusive and should not be used with other options.

`-m`
`--mark`

Tells FCU to set the boot select so that on the next boot the specified firmware bank will be active.

Currently, the mark operation only supports CPU firmware devices.

`-q`
`--query`

Tells FCU to return firmware information for a specific device (if used with -d) or information about all firmware devices. The query operation is exclusive and is not intended to be combined with other operations.

`-s`
`--show`

Display information about the target which the selected upgrade file corresponds to. Use this command to display the current firmware version status of the target device.

`-u`
`--upgrade`

Tells FCU to upgrade the currently inactive bank of the device specified by the target FRI file. The file option -f is required. The upgrade operation may be combined with the verify and mark operations.

`-v`
`--verify`

Tells FCU to verify the image file specified by the required -f option. This operation verifies that the specified file is sound and compatible with the current hardware. The verify operation may be combined with the upgrade and mark operations.

`--version`

Displays version information for the utility. This option is exclusive and should not be used with other options.

Usage

Some FCU options can be combined. Some options are exclusive. The following list describes the valid option combinations:

- --help
- --mark --bank=<bank-letter> --device=<device-id>
- --query
- --query --device=<device-id>
- --show --file=<filename>
- --upgrade --file=<filename>
- --verify --file=<filename>
- --verify --upgrade --file=<filename>
- --version

Multi character options may be abbreviated so long as they are unique. For example, `--upg` is equivalent to `--upgrade`. Typing **`--ver,`** however, will not work since it matches both `--verify` and `--version`.

Single-character options may be combined without repeating the hyphen, as in these examples:

- **fcu –vf <filename>**
- **fcu –q –d <device-id>**

Options are not case-sensitive. For example, `--help` is equivalent to `--HeLp`. However, option arguments, such as filename and device ID, are case-sensitive.

When upgrading firmware, it is strongly recommended that you upgrade only one device at a time. While FCU performs many checks during upgrade to ensure success, if something goes wrong and both firmware banks become corrupted, the module will be inoperable.

4.4 Upgrading a Firmware Image

This section describes recommended procedures for upgrading firmware devices. The procedures for upgrading BIOS and IPMC differ slightly.

4.4.1 BIOS Upgrade

The BIOS can only be upgraded from the module on which the BIOS is running. You have to upgrade the BIOS by using fcu. The hpmcmd command cannot be used to upgrade the BIOS.

Upgrading the BIOS for PrAMC-7311

Follow these steps to upgrade the BIOS.

1. Query the current BIOS firmware images on the module.
   fcu -qd <device-id>
   where <device-id> is the name of the device to be upgraded.
2. Verify the integrity of the upgrade file
   fcu --verify -f <path+filename>
   where <path+filename> is the complete path to the upgrade file, for example:
   fcu --verify -f /opt/bladeservices/rom/pramc-7311-cpu.fri
   Be sure the upgrade image is actually newer than the current firmware image.
3. Upgrade the firmware image
   fcu --upgrade -f <path+filename>
   where <path+filename> is the complete path to the upgrade file, for example:
   fcu --upgrade -f /opt/bladeservices/rom/pramc-7311-cpu.fri

   FCU writes the new image and then reads back the image and performs a binary compare to ensure that the write was successful. If the upgrade was not successful, you will see an error message. Try the upgrade again. If it is still not successful, contact your SMART EC representative.

4. Query the new image to ensure that the version information is correct
   fcu -qd <device-id>
   where <device-id> is the name of the upgraded device, for example:
   fcu -qd pramc-7311-cpu
5. Mark the new image as active so that it will be used for the next boot, for example:
   fcu --mark -b <bank-letter> -d <device-id>
   where <bank-letter> is the letter or number of the upgraded bank and <device-id> is the name of the upgraded device, for example:
   fcu --mark -b a -d pramc-7311-cpu
   or
   fcu --mark -b 0 -d pramc-7311-cpu

4.4.2 IPMC Upgrade

Upgrading IPMC for PrAMC-7311

Follow these steps to upgrade an IPMC:

1. Query the current IPMC firmware images on the module.
   fcu -qd <device-id>
   where <device-id> is the name of the device to be upgraded.
2. Verify the integrity of the upgrade file
   fcu --verify -f <path+filename>
   where <path+filename> is the complete path to the upgrade file, for example:
   fcu --verify -f /opt/bladeservices/rom/pramc-7311-ipmc.fri
   Be sure the upgrade image is actually newer than the current firmware image.
3. Upgrade the firmware image
   fcu --upgrade -f <path+filename>
   where <path+filename> is the complete path to the upgrade file, for example:
   fcu --upgrade -f /opt/bladeservices/rom/pramc-7311-ipmc.fri
   FCU writes the new image and then reads back the image and performs a binary compare to ensure that the write was successful. If the upgrade was not successful, you will see an error message. Try the upgrade again. If it is still not successful, contact your SMART EC representative.
4. Query the new image to ensure that the version information is correct,
   fcu -qd <device-id>
   where <device-id> is the name of the upgraded device, for example:
   fcu -qd H8S-AMCm F/W

If the version you just installed is now the active image, the upgrade was successful.

Hardware Platform Management

Hardware management is based on the Intelligent Platform Management Interface (IPMI) specification. IPMI commands can be complex and cumbersome. To facilitate module-level management, SMART EC provides the Hardware Platform Management (HPM) package that provides a set of commands that are based on IPMI commands but which are easier to use than the IPMI commands itself. An HPM command can encapsulate a sequence of IPMI commands. An HPM command can be the unifier for OEM IPMI commands that are different on different module types, for example reading the BIOS boot bank. For a catalogue of supported IPMI commands of the module refer to the respective *IPMI manual.*

Since HPM provides full access to the underlying IPMI subsystem, HPM commands should be used with special care. Improper usage of HPM commands may cause system corruption or system failure. Examples of improper usage are modifying Field Replaceable Unit (FRU) data, change blue led state, change e-keying, etc.

The HPM package consists of

- HPM daemon called hpmagentd
- Command line client called hpmcmd
- Script framework for managing shutdown and reboot events

The hpmcmd executes the incoming HPM commands and returns the result

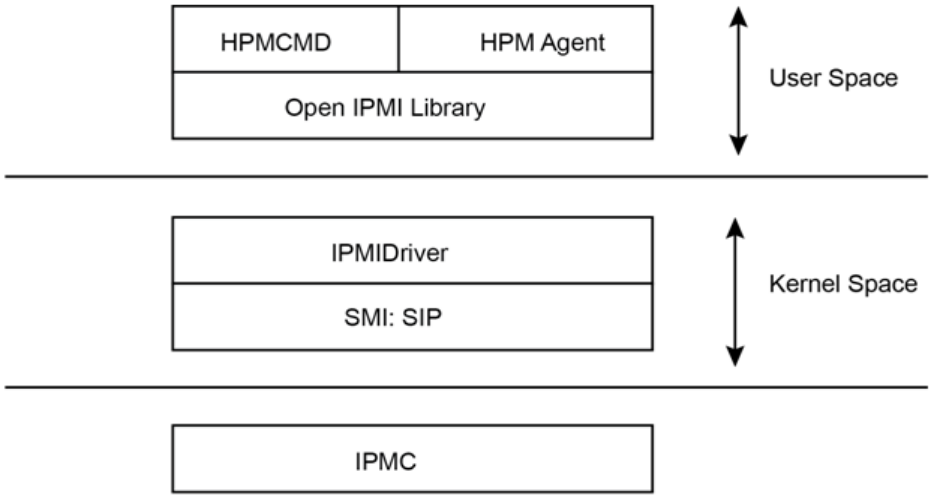
HPM commands include:

- Retrieving FRU data
- Reading and controlling status of IPMI-controlled LEDs
- Communicating local slot location information

HPM Agent executes shutdown and reboot scripts in response to shutdown or graceful reboot events received.

The hpmcmd and hpmagentd makes use of OpenIPMI to communicate to the local Intelligent Platform Management Controller (IPMC) using the System Management Interface (SMI). This SMI gets set up by the OpenIPMI driver. OpenIPMI consists of two main parts: A device driver that goes into the Linux kernel, and a user-level library. The following picture shows the software levels that are involved in the HPM architecture:

Figure 5-1 Software Levels of the HPM Architecture



SMI: System Management Interface
SIP: Serial Interface Protocol

The System Management Interface (SMI) driver provides the low level interface for communicating to IPMC. The communication is based on a serial interface protocol.

If you need more information about the software aspects of the blade and module IPMC, refer to the respective IPMI manual.

5.1 hpmagentd – HPM Agent Daemon

The HPM agent daemon handles local communication to the IPMC on a module using the SMI. This SMI gets set up by the OpenIPMI driver.

By default, the hpmagentd binary executable is installed in `/opt/bladeservices/bin/`directory. This directory has been added to the PATH environment variable.

This daemon has an init script called hpm that will start the daemon in run level 3 with the default settings.

When hpmagentd receives a graceful reboot or shutdown alert from the IPMC, it will call the respective script to run the reboot or shutdown sequence.

Synopsis

```
hpmagentd [-l log-level] [-r reboot-script] [-s shutdown-script]
```

```
hpmagentd {-i | -h | -v}
```

Parameters

`-l log-level`

Specifies the level of message logging, where log-level is one of the standard syslog levels:

Log Level	Description
0	Emergency
1	Alert
2	Critical
3	Error
4	Warning
5	Notice (default)
6	Information
7	Debug

`-r reboot-script`

Specifies a graceful reboot script that will be called when a module graceful reboot request is received by the Module Management Controller (MMC), where reboot-script is the complete path and filename of the target script. The default is `/opt/bladeservices/bin/hpmreboot` (see *hpm – Shutdown and Reboot Scripts on page 40*).

`-s shutdown-script`

Specifies a shutdown script that will be called when a module shutdown request is received by the MMC, where shutdown-script is the complete path and filename of the target script. The default is `/opt/bladeservices/bin/hpmshutdown` (see *hpm – Shutdown and Reboot Scripts on page 40*).

`-i`

hpmagentd runs interactively, that is it will not run as daemon.

`-h`

Displays a brief message about command usage.

`-v`

Displays the version of hpmagentd

5.2 hpm – Start-Up Script

An HPM agent init script, hpm, allows you to start, stop, and restart the HPM agent daemon using the agent's default option settings. By default, this script is installed in the `/opt/bladeservices/etc/init.d` directory during installation of the BBS software. It is also linked into the system start-up directories to automatically start the HPM agent when the system boots.

Synopsis

```
hpm {start | stop | restart | reload | status}
```

Parameters

`start`

Starts the hpm agent daemon.

`stop`

Terminates the hpm agent daemon.

`restart`

Terminates and then starts the hpm agent daemon.

`reload`

Terminates and then starts the hpm agent daemon.

5.3 hpm – Shutdown and Reboot Scripts

At any time during normal operation, a shelf manager may issue a shutdown (FRU Activation Deactivate) or graceful reboot (FRU Control Reboot) request to the MMC on a given module. The MMC then forwards this information to the HPM agent. The HPM agent listens for such requests from the MMC. When it receives a request, it calls the respective script to run the reboot or shutdown sequence. In case of a shutdown indication, all running processes should be notified about the shutdown. In case of a reboot notification, the payload is responsible for invoking the reboot procedure. The MMC is not involved in this process. This allows processes currently running on the module to prepare for shutdown. After the notification, it takes roughly 30 seconds before the payload is powered off.

Two default scripts, hpmshutdown and hpmreboot, are installed by default in the `/opt/bladeservices/bin` directory. Currently, these scripts simply print a banner indicating they have run and then issue `shutdown -h now` (hpmshutdown script) or `reboot` (hpmreboot script).

You may modify the default scripts to suit the needs of your system application or create new scripts. If you create new scripts, use the `-s` and `-r` options when starting hpmagentd to specify the new locations and names of the scripts. You may also need to update the hpm start up script in `/opt/bladeservices/etc/init.d/hpm`.

Synopsis

`hpmshutdown`

`hpmreboot`

5.4 hpmcmd – HPM Command Utility

The HPM command utility accepts commands from the user and executes them. Once a command is sent, the hpmcmd program waits until the answer from the IPMC is received or until a time-out occur.

The HPM command utility can be started in interactive mode, where a prompt is displayed and the user enters commands; it can read in a file of commands; or it can process a single command.

By default, the hpmcmd binary executable is installed in `/opt/bladeservices/bin`. During installation of the BBS software, this directory is added to the PATH environment variable.

If you do not provide any options you will see the following prompt once the program starts running:
`hpmcmd>`

From there you can start executing commands.

Synopsis

```
hpmcmd [options]
-- help

-c  process a single command
-h  displays this help message
-v  verbose mode for some commands
-t  send the command to a remote target
-f  file option used by some hpcmds
-p  change the prompt
```

Parameters

`-p new-prompt`

Specifies the prompt you would like to have for the hpmcmd interactive mode, where new-prompt is any string. The default prompt is **hpmcmd>**. This option should not be combined with the `-r` or -c options.

`-c command`

This option executes a single command and terminates, where command is one of the supported commands. This allows you to use the arrow history functions supported in the base shell; a history is not available inside the hpmcmd program. This option should not be combined with the -i option.

If this option is combined with `-o`, -c should be last option entered, since all arguments that follow `-c` on the command line will be considered part of the command.

5.4.1 Command Overview

The following table lists all commands from the hpmcmd program and shows for which blades/modules they are available. You can display this list and a short command description using the help command (see section *help on page 51*). A detailed description of the commands is given in section *Supported Commands on page 44*.

Table 5-1 Command Overview

Command	Description
bootbankget	Gets the bootbank to boot from
bootbankset	Sets the bootbank to boot from
chinfo	Retrieve channel information
cmd	Execute any IPMI command
deviceid	Gets the Device Id.
frudata	Allows to get FRU info in hex numbers
fruinfoget	Gets string fields from the FRU
fruinfoset	Sets string fields of the FRU
fruinv	Allows to get the FRU size and addressable units
fruread	Allows to read x number of bytes from the FRU

Table 5-1 Command Overview (continued)

Command	Description
fruwrite	Allows to write x number of bytes from the FRU
help	List of hpmcmd commands
ipmbaddress	Get the IPMB address
ipmcstatus	Get the IPMC status
lancfgget	Get LAN configuration parameter
lancfgset	Set LAN configuration parameter
ledget	Gets the state of a specific FRU LED
ledprop	Get the LED properties for this FRU
ledset	Controls the state of a specific FRU LED
loglevelget	Gets the hpmagentd log level
macaddress	Lists the MAC addresses
partnumber	Gets the board part number
physlotnumber	Lists the physical slot location
portget	Shows the current state E-Key governed intfs
portset	Enables/Disables ports in a channel
posttypeget	Gets the POST type to run at boot
posttypeset	Sets the POST type to run at boot
sdr	Shows the SDR records
sdr_dump	shows the SDR records in binary and hex format
sdrinfo	shows the SDR information
sendamc	Sends any IPMI command to a remote AMC or MMC of a remote IPMC IPMB-L
sendcmd	Sends an IPMI request to the IPMC

Table 5-1 Command Overview (continued)

Command	Description
shelfaddress	Gets the shelf address string
shelfslots	Prints the number of slots in the shelf
shelftype	Gets the Shelf Type from the Shelf FRU (Board Product Name)
slotmap	Prints the slotmap of the shelf
slotnumber	Shows the board logical slot number
solcfgget	Get SOL configuration parameter
solcfgset	Set SOL configuration parameter
version	Shows the hpmCmd version and the hpmagentd version
watchdog	Control Payload WDT functionality

5.4.2 Supported Commands

This section lists the supported commands. All commands are case insensitive. The examples illustrate the use of hpmcmd in single-command mode (-c). If you start hpmcmd without the -c or -i options (that is, interactive mode), you simply enter these commands at the HPM command prompt.

Some of the hpm commands can be sent to a remote IPMC by specifying the -t option. This option is not mandatory. If it is not specified, the command is sent to the local IPMC.

The following commands support the -t option:

- deviceid
- frudata
- fruinfoget
- fruinfoset
- fruinv
- fruread
- fruwrite
- lancfgget
- lancfgset

- ledget
- ledset
- macaddress
- partnumber
- portset
- solcfgget
- solcfgset

The following sections describe the available commands.

5.4.2.1 bootbankget

This command retrieves the boot bank which is currently marked as active for the CPU specified by payload_cpu_selector.

Firmware for the CPU is stored in redundant, persistent memory devices. This allows the firmware image in one bank to serve as a backup for the other bank. During normal operation, the CPU on a module determines which bank to boot from based on a General Purpose Input/Output (GPIO) signal controlled by the IPMC. This bank is considered the active boot device.

Because you can change the “active” device with the hpmcmd bootbankset command, active status does not necessarily indicate which device was used on the last boot. It simply represents which device is set to be used on the next boot.

Synopsis

bootbankget <payload_cpu_selector>

Parameters

`payload_cpu_selector`

Is an integer between 0 and the number of CPU devices supported on the module.

On the PrAMC-7311 the only valid value for payload_cpu_selector is 0.

Example

```
hpmcmd -c bootbankget 0
```

5.4.2.2 bootbankset

This command sets the boot bank for a particular CPU from which the module is supposed to boot.

Synopsis

```
bootbankset <payload_cpu_selector> <newBootBank>
```

Parameters

`tor`

Is an integer between 0 and the number of CPU devices supported on the module.

`newBootBank`

Can be set to BANK0 or BANK1

Example

```
hpmcmd -c bootbankset 0 BANK1
```

5.4.2.3 chinfo

This command allows you to retrieve the channel information.

Synopsis

```
chinfo <channel> [-t ipmb addr [:mmc addr]]
```

Parameters

`channel`

Channel number

`-t`

Sends the command to ipmb addr:mmc addr

Example

```
hpmcmd -c chinfo 0
```

5.4.2.4 cmd

This command allows you to enter commands understood by the IPMC. Commands are entered as a sequence of hexadecimal numbers as defined in the *IPMI 1.5 Specification*.

Synopsis

```
cmd <IPMI command>
```

Parameters

`IPMI command`

Sequence of hex bytes as entered using the ipmicmdtool from the OpenIPMI library.

```
cmd 0f 00 XX ZZ W1 W2 ... Wn)
where:
XX    = netfunct as it is in hex
ZZ    = cmd number as stated in the IPMI/PICMG spec
W1-Wn = data bytes according to what the command supports
```

Example

GetDeviceId command to the local IMPC:
`hpmcmd -c cmd f 0 6 1`

GetDeviceId command to the remote IPMC on address 7a:
`hpmcmd -c cmd 0 7a 0 6 1`

5.4.2.5 deviceid

This command retrieves the raw IPMI Get Device ID response and decodes the IPMI message.

Synopsis

```
deviceid -t [ipmbAddr[:mmcAddr]]
```

Parameters

`-t`

Sends the command to ipmbAddre:mmcAddr. ipmbAddr is the string lc if it is a local mmcAddr.

Example

`hpmcmd -c deviceid`

5.4.2.6 frudata

This command dumps the content of the FRU data in hexadecimal format.

Synopsis

```
frudata <fruid> [-t ipmbAddr[:mmcAddr]]
```

Parameters

`fruid`

Is 0 for the main module.

`-t`

Sends the command to ipmbAddr:mmcAddr. ipmbAddr is the string lc if it is a local mmcAddr.

Example

```
hpmcmd -c frudata 0
hpmcmd -c frudata 0 -t 20
```

5.4.2.7 fruinfoget

This command retrieves information from the specified FRU.

Synopsis

```
fruinfoget <fruid> [field] [-v] [-t ipmbAddr[:mmcAddr]]
```

Parameters

`fruid`

0 for the Main Board FRU and other for the RTM or AMC FRU.

`field`

Is one of the following data fields. If no field is specified, it retrieves the whole fruinfo for that FRU.

Field	Description
bmanufacturer	Board manufacturer
bproductname	Board product name
bserialnumber	Board serial number
bpartnumber	Board part number
pmanufacturer	Product manufacturer
pproductname	Product product name
ppartnumber	Product part number
pversion	Product version number
pserialnumber	Product serial number
passettag	Product inventory asset identifier

`-v`

Verbose mode to get point-to-point connectivity information where no specific field is requested.

`-t`

Sends the command to ipmbAddr:mmcAddr. ipmbAddr is the string lc if it is a local mmcAddr.

Example

```
hpmcmd -c fruinfoget 1 bmanufacturer
```

The following example for fruinfoget is without fields and -v option.

```
hpmcmd -c fruinfoget 0
```

5.4.2.8 fruinfoset

This command sets some individual field in a FRU, or replaces the whole content of the FRU from a file.

Synopsis

```
fruinfoset <fruid> -f <frufilepath> [-t ipmbAddr[:mmcAddr]]
fruinfoset <fruid> [field] <newvalue> [-t ipmbAddr[:mmcAddr]]
```

Parameters

`fruid`

0 for the Main Board FRU and other for the RTM or AMC FRU.

`newvalue`

It is the new value, less than 16 bytes long, to be set.

`frufilepath`

It is the full FRU info binary file path, when using a file.

`field`

Is one of the following data fields. If no field is specified, it retrieves the whole fruinfo for that FRU.

Field	Description
bmanufacturer	Board manufacturer
bproductname	Board product name
bserialnumber	Board serial number
bpartnumber	Board part number
pmanufacturer	Product manufacturer
pproductname	Product product name
ppartnumber	Product part number
pversion	Product version number
pserialnumber	Product serial number
passettag	Product inventory asset identifier

`-t`

Sends the command to ipmbAddr:mmcAddr. ipmbAddr is the string lc if it is a local mmcAddr.

Example

```
hpmcmd -c fruinfoset 1 bmanufacturer Artesyn
```

5.4.2.9 fruinv

This command retrieves the FRU size and the addressable unit for the specified FRU.

Synopsis

```
fruinv <fruid> [-t ipmbAddr[:mmcAddr]]
```

Parameters

`fruid`

0 for the Main Board FRU and other for the RTM or AMC FRU.

`-t`

Sends the command to ipmbAddr:mmcAddr. ipmbAddr is the string lc if it is a local mmcAddr.

Example

```
hpmcmd -c fruinv 0
```

5.4.2.10 fruread

This command gets nBytes of FRU fruid from the startAddress in hex notation.

Synopsis

```
fruread <fruid> <startAddress> <nBytes> [-t ipmbAddr [:mmcAddr]]
```

Parameters

`fruid`

0 for the Main Board FRU and other for the RTM or AMC FRU.

`startAddress`

The starting point from where to read.

`nbytes`

Number of bytes to read in decimal; cannot exceed 16 because of IPMI message size limitations.

`-t`

Sends the command to ipmbAddr:mmcAddr.

Example

```
hpmcmd -c fruread 0 0 10
hpmcmd -c fruread 0 0 10 -t 20
```

5.4.2.11 fruwrite

This command writes hex byte values to FRU fruid starting at the startAddr in hex notation.

Synopsis

`fruwrite <fruid> <startAddress> <hexval1> [hexval2] [...] [hexval16] [-t ipmbAddr [:mmcAddr]]`

Parameters

`fruid`

0 for the Main Board FRU and other for the RTM or AMC FRU.

`startAddress`

The starting point from where to write.

`hexvalN`

Hex byte value to write.

`-t`

Sends the command to ipmbAddr:mmcAddr.

5.4.2.12 help

This command lists the available commands from the hpmcmd program with a brief explanation about the command.

Synopsis

`help`

5.4.2.13 ipmbaddress

This command retreives the module IPMB address.

Synopsis

`ipmbaddress`

Example

hpmcmd -c ipmbaddress

5.4.2.14 ipmcstatus

This command retrieves the module ipmc status.

Synopsis

ipmcstatus

Example

hpmcmd -c ipmcstatus

5.4.2.15 lancfgget

This command gets LAN configuration parameter.

Synopsis

lancfgget <channel> [param] [-t ipmbAddr[:mmcAddr]]

Parameters

channel

channel number

param

auth-type-support

auth-type-enables

ip-addr

ip-addr-src

mac-addr

subnet-mask

ipv4-header-params

primary-rmcp-port

secondary-rmcp-port

bmc-generated-arp-control

gratuidous-arp-interval

default-gateway-addr

default-geteway-mac-addr

backup-gateway-addr

backup-geteway-mac-addr
community-string
num-destinations
destination-type
destination-addr
vlan-id
vlan-prio
rmcp-cipher-support
rmcp-ciphers
rmcp-priv-levels
dst-addr-vlan-tags

`-t`

sends the command to ipmbAddr:mmcAddr.

Example

```
hpmcmd -c lancfgget 1
```

5.4.2.16 lancfgset

This command sets LAN configuration parameter.

Synopsis

```
lancfgset <channel> <param> <value> [-t ipmb_addr[:mmc_addr]]
```

Parameters

`channel`

channel number

`param`

ip-addr
subnet-mask
default-gateway-addr

`value`

IP address in string format.

Example

```
hpmcmd -c lancfgset 1 ip-addr 172.16.0.222
```

5.4.2.17 ledget

This command gets information about a specified LED controlled by the IPMC.

Synopsis

```
ledget <fruid> <led> [-t ipmbAddr[:mmcAddr]]
```

Parameters

`fruid`

FRUID

`led`

BLUE (for the hotswap LED)
LEDN (for FRU led N, n is a number between 1 and the maximum number of FRULEDs supported by the board)

`-t`

Sends the command to ipmbAddr:mmcAddr. ipmbAddr is the string lc if it is a local mmcAddr.

Example

```
hpmcmd -c ledget 0 BLUE
hpmcmd -c ledget 0 led1
```

5.4.2.18 ledprop

This command lists the LEDs in this FRU controlled by the IPMC.

Synopsis

```
ledprop <fruid>
```

Parameters

`fruid`

0 for the Main Board FRU and 1 for the RTM FRU.

Example

```
hpmcmd -c ledprop 0
```

5.4.2.19 ledset

This command controls the override state of a specific FRU LED. The RTM FRU LEDs reflect the state of the main blade (FRU 0) LEDs. Therefore, overriding the state to something different than the main FRU LED state will not have any effect.

The blue LED is the only one that can be controlled separately.

Synopsis

```
lledset <fruid> <led> OFF [-t ipmb_addr[:mmc_addr]]
ledset <fruid> <led> LOCAL [-t ipmb_addr[:mmc_addr]]
ledset <fruid> <led> ON [color] [-t ipmb_addr[:mmc_addr]]
ledset <fruid> <led> TEST [duration] [-t ipmb_addr[:mmc_addr]]
ledset <fruid> <led> BLINK [offms] [onms] [color] [-t
ipmb_addr[:mmc_addr]]
```

Parameters

`fruid`

0 for the front board. Other for RTMs or AMCs.

`led`

BLUE (for the hotswap LED)
LEDN (for FRU led N, n is a number between 0 and the maximum number of FRU LEDs supported by the board)

`operation`

ON (for turning it on)
OFF (for turning it off)
BLINK (for blinking (dft 300ms/300ms))
LOCAL (for return to local control)
TEST (for running lamptest (dft 5000ms))

`offms`

Off duration when blinking; Values from 10ms to 2500ms in 10ms increments.

`onms`

On duration when blinking; Values from 10ms to 2500ms in 10ms increments.

`duration`

lamptest duration. values from 100ms to 12800ms in increments of 100ms should be used.

`color`

LED0 = BLUE
LED1 = RED or AMBER
LED2 = GREEN (if supported by IPMC)
LED3 = AMBER (if supported by IPMC)

`-t`

Sends the command to ipmb_addr:mmc_addr

Example

```
hpmcmd -c ledset 0 led1 on
```

5.4.2.20 loglevelget

This command retrieves the current hpm log level.

Synopsis

```
loglevelget
```

Example

```
hpmcmd -c loglevelget
```

5.4.2.21 macaddress

This command retrieves a list of available MAC addresses.

Synopsis

```
macaddress [-t ipmbAddr]
```

Parameters

`-t ipmbAddr`

Sends the command to ipmbAddr.

Example

```
hpmcmd -c macaddress
hpmcmd -c macaddress -t 20
```

5.4.2.22 partnumber

This command retrieves the part number of the main module.

Synopsis

```
partnumber [-t ipmbAddr]
```

Parameters

`-t ipmbAddr`

Sends the command to ipmbAddr.

Example

```
hpmcmd -c partnumber
hpmcmd -c partnumber - t 20
```

5.4.2.23 physlotnumber

This command retrieves the physical slot number in which the blade is plugged in. It is only applicable to AdvancedTCA system environments.

Synopsis

`physlotnumber`

Example

`hpmcmd -c physlotnumber`

5.4.2.24 portget

This command shows the current state of interfaces governed by e-keying. If no channel is specified, portget returns data for all channels in the specified interface. If neither interface nor channel are specified, portget will return data for all interfaces.

Synopsis

`portget [interface] [channel]`

Parameters

`interface`

The only valid value for PrAMC-7311 is AMC.

`channel`

Is an integer in the following range:
0 and 1 for Ethernet links for Base
2 and 3 for SATA storage links
4 and 5 for PCI-Express links

Example

`hpmcmd -c portget`
`hpmcmd -c portget AMC 0`

5.4.2.25 portset

This command enables and disables ports in a channel. The following table lists the valid values for each parameter.

Synopsis

`portset <intf> <chan> <grpid> <type> <typeX> <ports> <oper> [devid] [-t ipmbAddr[:mmcAddr]]`

Parameters

`intf`

Valid value for the PrAMC-7311 is: AMC

chan

Is an integer in the following range:
0 and 1 for Ethernet links for Base
2 and 3 for SATA storage links
4 and 5 for PCI-Express links

`grpid`

0 for Ethernet links and SATA storage links
0 or 2 for PCI-Express links

`type`

Valid values are:

ETHER for fabric interface
STORAGE for storage links

`ports`

0 for Ethernet links
0123 for PCI-Express links

`oper`

Valid values are DISABLE or ENABLE.

Example

```
hpmcmd -c portset base 1 0 base 0 0 enable
```

5.4.2.26 posttypeget

This command retrieves the post_type to which the board is currently set to run at boot time for the particular CPU specified.

Synopsis

```
posttypeget <payload_cpu_selector>
```

Parameters

`payload_cpu_selector`

The particular CPU specified is set to posttype to run.

Example

```
hpmcmd -c posttypeget 0
```

5.4.2.27 posttypeset

This command sets the posttype to which the board is currently set to run at boot time for theparticular CPU specified.

Synopsis

```
posttypeset <payload_cpu_selector> <newPostType>
```

Parameters

`payload_cpu_selector`

Integer between 0 and number of CPU devices supported per board.

`newPostType`

POST type. Supported values are: SHORT and LONG

Example

```
hpmcmd -c posttypeset 0 LONG
```

5.4.2.28 sdr

This command shows the SDR records.

Synopsis

```
sdr
```

Example

```
hpmcmd -c sdr
recID 1: management controller device locator record
    I2C slave addr:  3D
    Channel number:  00
    Power state:     06
    Global init:     0C
    Capabilities:    29
    Entity Id:       AMC
    Entity instance: 65
    OEM:             00
    Id string:       PrAMC-731x

recID 2: full sensor record
    owner is IPMB 7A sensor num 00 on lun 00 channel 00
```

```
    logical entity: AMC - instance 65
    Module Hot Swap: module hot swap: sensor-specific discrete

recID 3: full sensor record
    owner is IPMB 84 sensor num 02 on lun 00 channel 00
    logical entity: AMC - instance 65
    MP +3.3V: voltage; threshold

recID 4: full sensor record
    owner is IPMB 7A sensor num 02 on lun 00 channel 00
    logical entity: AMC - instance 65
    +12V: voltage: threshold
recID 5: full sensor record
    owner is IPMB 7A sensor num 03 on lun 00 channel 00
    logical entity: AMC - instance 65
    UCD3V3: voltage: threshold
recID 6: full sensor record
    owner is IPMB 7A sensor num 04 on lun 00 channel 00
    logical entity: AMC - instance 65
    VCCCORE: voltage: threshold

recID 7: full sensor record
    owner is IPMB 7A sensor num 05 on lun 00 channel 00
    logical entity: AMC - instance 65
    MMC Health : management subsystem health : sensor-specific discrete
recID 8: full sensor record
    owner is IPMB 7A sensor num 06 on lun 00 channel 00
    logical entity: AMC - instance 65
    Version Change: 2B: sensor-specific discrete
recID 9: full sensor record
    owner is IPMB 7A sensor num 07 on lun 00 channel 00
    logical entity: AMC - instance 65
    BMC Watchdog : watchdog 2 : sensor-specific discrete
recID 10: full sensor record
    owner is IPMB 7A sensor num 08 on lun 00 channel 00
    logical entity: AMC - instance 65
```

```
    Processor : processor : sensor-specific discrete
recID 11: full sensor record
    owner is IPMB 7A sensor num 09 on lun 00 channel 00
    logical entity: AMC - instance 65
    ALLPGOOD : power supply : generic
recID 12: full sensor record
    owner is IPMB 7A sensor num 0A on lun 00 channel 00
    logical entity: AMC - instance 65
    F/W Progress : system firmware progress : sensor-specific discrete
recID 13: full sensor record
    owner is IPMB 7A sensor num 0B on lun 00 channel 00
    logical entity: AMC - instance 65
    PCH Temp : temperature : threshold
recID 14: full sensor record
    owner is IPMB 7A sensor num 0C on lun 00 channel 00
    logical entity: AMC - instance 65
    Inlet Temp : temperature : threshold
recID 15: full sensor record
    owner is IPMB 7A sensor num 0D on lun 00 channel 00
    logical entity: AMC - instance 65
    Power Temp : temperature : threshold
recID 16: full sensor record
    owner is IPMB 7A sensor num 0E on lun 00 channel 00
    logical entity: AMC - instance 65
    Ethernet Temp : temperature : threshold
recID 17: full sensor record
    owner is IPMB 7A sensor num 0F on lun 00 channel 00
    logical entity: AMC - instance 65
    CPU Temp : temperature : threshold
recID 18: full sensor record
    owner is IPMB 7A sensor num 10 on lun 00 channel 00
    logical entity: AMC - instance 65
    Boot Error : boot error : sensor-specific discrete
recID 19: full sensor record
    owner is IPMB 7A sensor num 11 on lun 00 channel 00
    logical entity: AMC - instance 65
```

```
    OS Boot : os boot : sensor-specific discrete
recID 20: full sensor record
    owner is IPMB 7A sensor num 12 on lun 00 channel 00
    logical entity: AMC - instance 65
    Boot Bank : D2 : sensor-specific discrete
```

5.4.2.29 sdr_dump

This command shows the SDR records in binary and hex format.

Synopsis

```
sdr_dump
```

Example

```
hpmcmd -c sdr_dump
SDR Records:
01 00 51 01 39 20 00 10 14 61 7f 69 02 01 04 22 "..Q.9 ...a.i..."
04 22 12 12 00 04 00 00 33 00 00 00 00 c0 07 cd "."......3......Í"
d0 ca ff 00 00 d8 00 00 c2 00 01 01 00 00 00 ce "..............Î"
53 42 43 20 2b 31 2e 30 35 56 20 56 74 74 "SBC +1.05V Vtt"
.
.
.
61 67 65 20 45 "age E"
```

5.4.2.30 sdrinfo

This command shows the SDR information.

Synopsis

```
sdrinfo
```

Example

```
hpmcmd -c sdrinfo
SDR Information:
LUN 0 has 019 sensors; static sensor population
LUN 1 has 000 sensors; static sensor population
LUN 2 has 000 sensors; static sensor population
LUN 3 has 000 sensors; static sensor population
```

5.4.2.31 sendamc

This command allows sending any of the commands supported in the IPMI spec to a remote AMC or MMC of a remote IPMC IPMB-L.

Synopsis

```
sendamc <IPMBaddress> <MMCaddress> <netfn> <cmd> <data0> ... <datan>
```

Parameters

`IPMBaddress`

Destination IPMB address in hex digits.

`MMCaddress`

Destination MMC address in hex digits.

`netfn`

IPMI request net function in hex digits.

`cmd`

IPMI request command in hex digits.

`data0-datan`

IPMI request data bytes. if any, in hex digits.

5.4.2.32 sendcmd

This command allows a user to send any of the commands supported in the IPMI spec to a remote IPMC.

Synopsis

```
sendcmd <IPMBaddress> <netfn> <cmd> <data0> ... <dataN>
```

Parameters

`IPMBaddress`

Destination IPMB address in hex digits.

`netfn`

IPMI request net function in hex digits.

`cmd`

IPMI request command in hex digits

`data0 ... dataN`

IPMI request data bytes. if any, in hex digits.

Example

```
hpmcmd -c sendcmd 7a 0a 11 00 00 00 10
Completion code: 0x00 (0) Success
Response data  : 10 01 00 00 01 0A 13 00 E1 01 09 19 E1 07 88 C7 45
hpmcmd -c sendcmd 20 0a 11 00 00 00 10
Completion code: 0x00 (0) Success
Response data  : 10 01 00 00 01 09 11 00 E4 01 08 19 B9 8F 87 C7 45
```

5.4.2.33 shelfaddress

This command retrieves the shelf address string from the shelf FRU.

Synopsis

```
shelfaddress
```

Example

```
hpmcmd -c shelfaddress
3
```

5.4.2.34 shelfslots

This command retrieves the number of slots in the shelf.

Synopsis

```
shelfslots
```

Example

```
hpmcmd -c shelfslots
14 slots
```

5.4.2.35 shelftype

This command retrieves the shelf type from the shelf FRU (Board Product Name)

Synopsis

```
shelftype
```

Example

```
hpmcmd -c shelftype
AXP-1440
```

5.4.2.36 slotmap

This command retrieves the slotmap of the shelf.

Synopsis

```
slotmap
```

Example

```
hpmcmd -c slotmap
-----------------------------------------------------------------
Physical Slot : 01 02 03 04  . 05 06 07 08  . 09 10 11 12  . 13 14
Logical  Slot : 13 11 09 07  . 05 01 03 04  . 02 06 08 10  . 12 14
IPMB  Address : 9A 96 92 8E  . 8A 82 86 88  . 84 8C 90 94  . 98 9C
-----------------------------------------------------------------
```

5.4.2.37 slotnumber

This command retrieves the logical slot number of the slot where the module is plugged in.

Synopsis

```
slotnumber
```

Example

```
hpmcmd -c slotnumber
2
```

5.4.2.38 solcfgget

This command retrieves the SOL configuration parameter.

Synopsis

```
solcfgget <channel> [param] [-t ipmbAddr[:mmcAddr]]
```

Parameters

channel

channel number

param

enable
authentication
char-settings
retry

nonvolatile-bit-rate
volatile-bit-rate
payload-channel
payload-port

`-t`

sends the command to ipmbAddr:mmcAddr

Example

```
hpmcmd -c solcfgget 1 retry
Retry Count                       : 1
Retry Interval                (ms): 50
hpmcmd -c solcfgget 0
Enabled                           : true
Force Encryption                  : false
Force Authentication              : false
Privilege Level                   : User
Character Accumulate Interval (ms): 5
Character Send Threshold          : 1
Retry Count                       : 1
Retry Interval                (ms): 50
Non-Volatile Bit Rate        (kps): 115.2
Volatile Bit Rate            (kps): 115.2
Payload Port                      : 623
```

5.4.2.39 solcfgset

This command retrieves the SOL configuration parameter.

Synopsis

```
solcfgset <channel> <param> <value> [-t ipmbAddr[:mmcAddr]]
```

Parameters

`channel`

channel number

`param`

force-encryption true|false
force-authentication true|false
privilege-level user|operator|administrator|oem
char-accumulate-interval 1-1275 (ms)
char-send-threshold 0-255

retry-count 0-255
retry-interval 0-2550 (ms)
non-volatile-bitrate 9.6|19.2|38.4|57.6|115.2
volatile-bitrate 9.6|19.2|38.4|57.6|115.2
port 0-255

`-t`

sends the command to ipmbAddr:mmcAddr

5.4.2.40 version

This command retrieves the version of the hpmcmd software and sends a request to get the version of the hpmagent daemon that is running. Once the information is gathered, it is printed.

Synopsis

```
version
```

Example

```
hpmcmd -c version
3.10.5
```

5.4.2.41 watchdog

This command is used handle the payload BMC watchdog.

Synopsis

```
watchdog set <tmr_use> <tmr_action> <pre_timeout> <flags> <lsb_val>
<msb_val>
watchdog set default
watchdog get
watchdog start
watchdog stop
watchdog reset
```

Parameters

`set`

Name	Possible values
tmr_use	dont_stop stop
tmr_action	no_action hard_reset power_cycle power_down
pre_timeout	0-255
flags	clear dont_clear
lsb_val	0-255
msb_val	0-255
user	

Example

```
hpmcmd -c watchdog get
hpmcmd -c watchdog set stop no_action 5 clear 100 1
hpmcmd -c watchdog stop
hpmcmd -c watchdog start
```

HPI-B Software

6.1 HPI-B Software

To ease the implementation of highly available systems with off-the-shelf building blocks, the Service Availability Forum (SA Forum) Hardware Platform Interface (HPI) specification HPIB defines a set of platform-independent programming interfaces to monitor and control systems, such as AdvancedTCA systems, designed to provide high availability. HPI provides applications and middle ware a consistent, standardized interface for managing hardware components.

6.2 Package Information

This BBS release contains the following HPI-B packages.

Table 6-1 List of HPI-B packages

File	Description
`bbs-hpib_<version>_amd64.deb`	This debian package contains the shared libraries to be used by HPI-B clients and some example HPI-B client applications.
`bbs-hpib-devel_<version>_amd64.deb`	This debian package contains the HPI-B header files and static libraries used for the development of HPI-B clients. This is part of host development toolkit.
`bbs-hpib-clientsrc_<version>_amd64.deb`	This debian package contains client sources as part of devkit for the application. This is part of host development toolkit.

6.3 Configuring the HPI-B client

HPI-B client software comprises of the packages listed in *Table 6-1*.

It consists of files required to develop your own client applications and also pre-compiled example applications. The counterpart of the HPI-B clients are the HPI-B daemons which run on the system manager blades (in our case, F125/F140). To configure HPI-B client, refer to *System Management Interface Based on HPI-B 2.0 User's Guide*.

Related Documentation

A.1 SMART Embedded Computing Documentation

The documentation listed is referenced in this manual. Technical documentation can be found by using the Documentation Search at *https://www.smartembedded.com/ec/support/* or you can obtain electronic copies of SMART EC documentation by contacting your local sales representative.

Table A-1 SMART EC Documentation

Document Title	Publication Number
System Management Interface Based on BBS HPI-B 2.0 (Centellis 4620/4440)	6806800P21
PrAMC-7311 Installation and Use	6806800P34
ATCA-F125 Installation and Use	6806800J94
ATCA-F120 Installation and Use	6806800D06
ATCA-F140 Installation and Use	6806800M67

A.2 Related Specifications

For additional information, refer to the following table for related specifications. As an additional help, a source for the listed document is provided. Please note that, while these sources have been verified, the information is subject to change without notice.

Table A-2 Related Specifications

Document Title	Source
IPMI Specifications http://www.intel.com/design/servers/ipmi	
IPMI Spec v1.5, Document Revision 1.1, February 20, 2002	Intel Corporation, Hewlett-Packard, DEC, NEC
IPMI v1.5 Addenda, Errata, and Clarifications, Addendum Document Revision 5, January 29, 2004	Intel Corporation, Hewlett-Packard, DEC, NEC
Intelligent Platform Management Interface Specification v1.0, Document Revision 1.1, November 15 1999	Intel Corporation, Hewlett-Packard, NEC, Dell
IPMI Implementer's Guide, Draft Version 0.7, September 16, 1998	Intel Corporation

Table A-2 Related Specifications (continued)

Document Title	Source
IPMI Platform Management FRU Information Storage Definition V1.0, September 27, 1999	Intel Corporation
PCI Industrial Computer Manufacturers Group (PICMG) Specifications http://www.picmg.org	
PICMG 3.0 Revision 1.0 Advanced Telecommunications Computing Architecture (AdvancedTCA) Base Specification, December 2002	PICMG
PICMG 3.1 Revision 1.0 Specification Ethernet/Fiber Channel for AdvancedTCA Systems, January 2003	PICMG
Service Availability Forum Specifications http://www.saforum.org	
SAI-HPI-B.01.01 Hardware Platform Interface Specification	SA Forum
SAI-AIS-A.01.01 Application Interface Specification	SA Forum
SAI-HPI-SNMP-B.01.01	SA Forum
SAIM-HPI-B.01.01-ATCA SAF HPI-to-AdvancedTCA Mapping Specification	SA Forum

A.3 Additional Resources

The following table lists additional resources which may be useful in working with SMART EC's AdvancedTCA systems.

Table A-3 Additional Resources

Resource	Source
OpenHPI open source software project http://openhpi.org	
OpenHPI 1.0 Manual	OpenHPI
OpenHPI NetSNMP Subagent Development Manual	OpenHPI
Net-SNMP http://net-snmp.sourceforge.net/	
Pigeon Point Systems http://www.pigeonpoint.com	

Table A-3 Additional Resources (continued)

Resource	Source
IPM Sentry Shelf-External Interface Reference	Pigeon Point Systems
IPM Sentry Shelf Manager User Guide	Pigeon Point Systems
OpenIPMI http://openipmi.sourceforge.net/	
Initramfs http://en.gentoo-wiki.com/wiki/Initramfs	

Related Documentation